Checksums

Using DDT

Author: William Cooke (DRC/HPTg)
Phone: +1 - 609 - 987 - 5054
Fax: +1 - 609 - 987 - 5063
Email: William.Cooke@noaa.gov

National Oceanic and Atmospheric Administration
Geophysical Fluid Dynamics Laboratory
Princeton, NJ 08542
http://www.gfdl.noaa.gov

Outline

- Overview
- Checksums
- ddt

Outline

- Overview
- Checksums
- ddt

Overview

- Given an array of numbers, can we expect a simple summation to represent the true value of every element of the array?
- Is there a way to quickly tell when values change?
- Is there a tool to show where the code is failing?

Outline

- Overview
- Checksums
- ddt

What is a checksum?

http://wordnetweb.princeton.edu/perl/webwn?s=checksum

Noun

S: (n) checksum (a digit representing the sum of the digits in an instance of digital data; used to check whether errors have occurred in transmission or storage)

Precision of numbers

Given a model of number representation, how different in magnitude can numbers be before one of the numbers is considered irrelevant?

Debug session

- Open a debug session
  - msub -X -q debug -I -l partition=t1,size=NPES, walltime=HH:MM:SS

  -X allows X-windows to display back

  -I = interactive

  -l = following are submit commands
- Do this in a freenx window for X response times.

Precision exercise setup

- msub -X -q debug -I -l partition=t1,size=32, walltime=02:00:00
- cd $CSCRATCH/$USER
- mkdir DDT
- cd DDT
- tar -xf ~William.Cooke/ddt_examples.tar

Precision exercise

- cd precision
- Inspect the code precision.F90
  - Simple code that adds 1e-n to a scalar.
  - Prints out running total of the scalar
- Exercise
  - ./run_precision_r8.csh
  - ./run_precision.csh

Results of adding 10^{-n}

10^ -0	1.0000000000000000000	10^-10	1.1111111111000000484
10^-1	1.1000000000000000888	10^-11	1.1111111111100000493
10^-2	1.1100000000000000977	10^-12	1.1111111111110001382
10^-3	1.1109999999999999876	10^-13	1.1111111111111000582
10^-4	1.1110999999999999766	10^-14	1.1111111111111100502
10^-5	1.1111100000000000421	10^-15	1.1111111111111111605
10^-6	1.1111109999999999598	10^-16	1.1111111111111111605
10^-7	1.1111111000000000182	10^-17	1.1111111111111111605
10^-8	1.1111111099999999574	10^-18	1.1111111111111111605
10^-9	1.1111111110000000402	10^-19	1.1111111111111111605

	Adding 1e-15		Adding 1e-16
1	1.0000000000000011	1	1.0000000000000000
2	1.0000000000000022	2	1.0000000000000000
3	1.0000000000000033	3	1.0000000000000000
4	1.0000000000000044	4	1.0000000000000000
5	1.0000000000000056	5	1.0000000000000000
6	1.0000000000000067	6	1.0000000000000000
7	1.0000000000000078	7	1.0000000000000000
8	1.0000000000000089	8	1.0000000000000000
9	1.0000000000000100	9	1.0000000000000000
10	1.0000000000000111	10	1.0000000000000000

Model of number format

http://en.wikipedia.org/wiki/Computer_number_format

IEEE 754-2008 standard defines a 64 bit floating point format with

11-bit exponent

52-bit significand

A sign bit

Real*8 (float) number =

(sign)*(1+<fractional significand>) * 2^(exponent-1023)

Model of number format

IEEE 754-2008 allows non zero number

- ±1.797693134862231E+308
- ±4.940656458412465E-324

Only good to 15 decimal places.

64 bits can be represented by 16 hexadecimal digits.

e.g. 1023(base10) = 11-1111-1111 (binary) = 3FF

1.0 = sign*(1+significand=0) * 2 ^(exponent=1023 -1023)

So 1.0 = 3FF0 0000 0000 0000

Model of number format

We still have the issue of trying to add 2 numbers together that are of vastly different magnitude.

Do this via the TRANSFER function.

- Use a long-integer (I*8)
- Bitwise representation of the argument.
- Do an integer sum over the elements of an array.

Checksums in FMS models

Note that if 2 elements of an array are switched the checksum will be identical. However as climate models are chaotic, a small change due to a bug will cause answers to diverge.

These divergences can be spotted using the checksum capability of FMS.

&coupler_nml : do_chksum=.true.

This prints out the checksums of various types before and after a subroutine is called in the main loop.

Comparison of output will allow you to narrow down where the divergence is occurring.

NB : The output is verbose. Use only if absolutely necessary.

Outline

- Overview
- Checksums
- ddt

What is ddt?

- ddt is a graphical debugger.
- Similar to totalview.
- Compile code with '-g -O0 '
- Do you really need a graphical debugger?
  - Maybe not. Examine the output of the program when run.

How to use?

Run in a directory that is visible to batch node

- /lustre/fs (Not /lustre/ltfs!)

Load the ddt module

- module load ddt
- ddt ./your_executable.x
- ddt -n $npes ./your_executable.x
- You don't need to add aprun.

HIGH PERFORMANCE TECHNOLOGIES GROUP

CSTARTMPI exercise

```
cd cstartmpi
./run_cstartmpi.csh
```

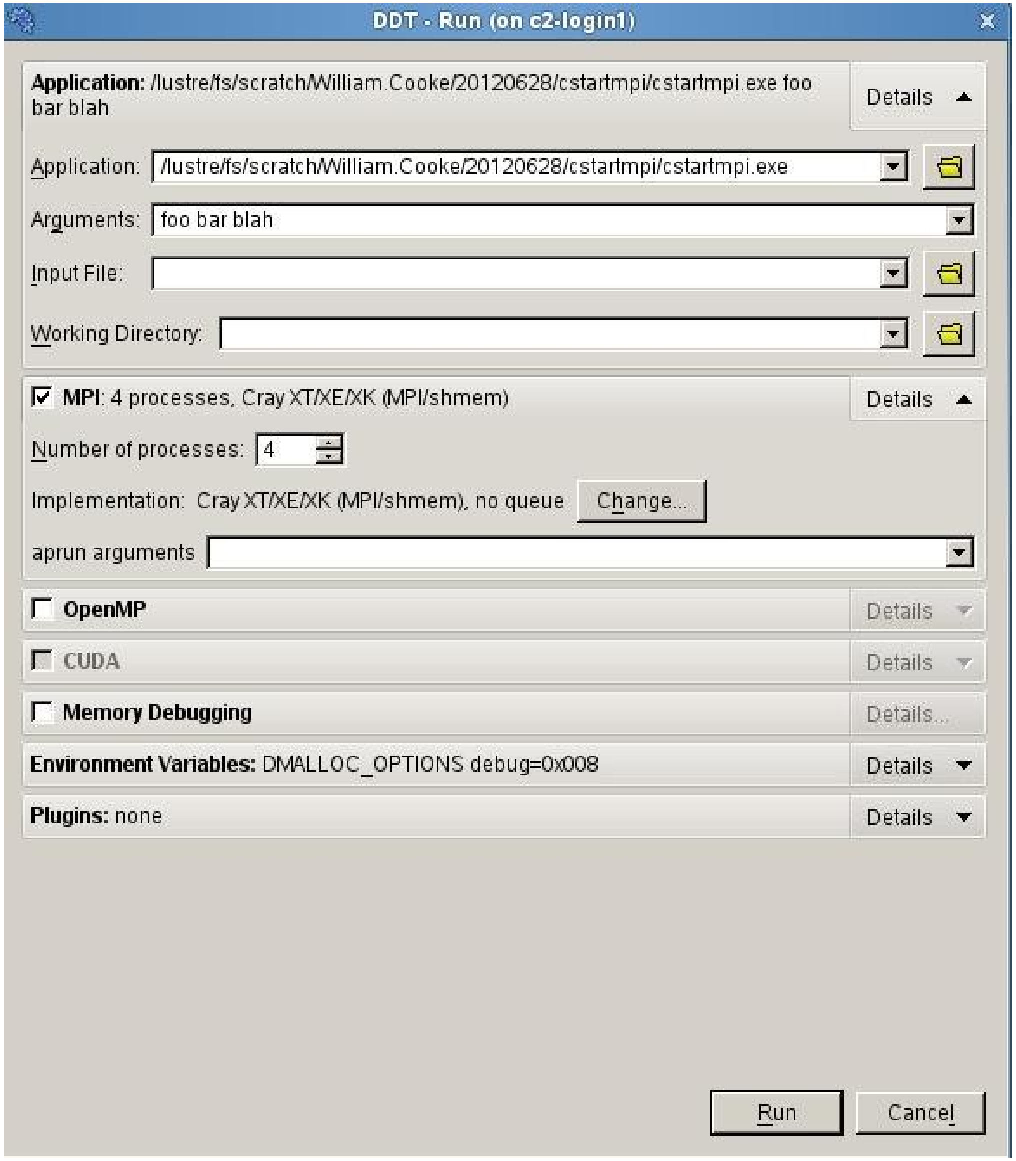
Runs program cstartmpi.exe

- without arguments (runs to completion)
- with arguments (fails)
- without arguments but a different pe count (fails with Segmentation fault)

Opening Window

You may change arguments.

Press run.



DDT - Run (on c2-login1)

Application: /lustre/fs/scratch/William.Cooke/20120628/cstartmpi/cstartmpi.exe foo bar blah

Application: /lustre/fs/scratch/William.Cooke/20120628/cstartmpi/cstartmpi.exe

Arguments: foo bar blah

Input File:

Working Directory:

☑ **MPI**: 4 processes, Cray XT/XE/XK (MPI/shmem)

Number of processes: 4

Implementation: Cray XT/XE/XK (MPI/shmem), no queue

aprun arguments

☐ **OpenMP**

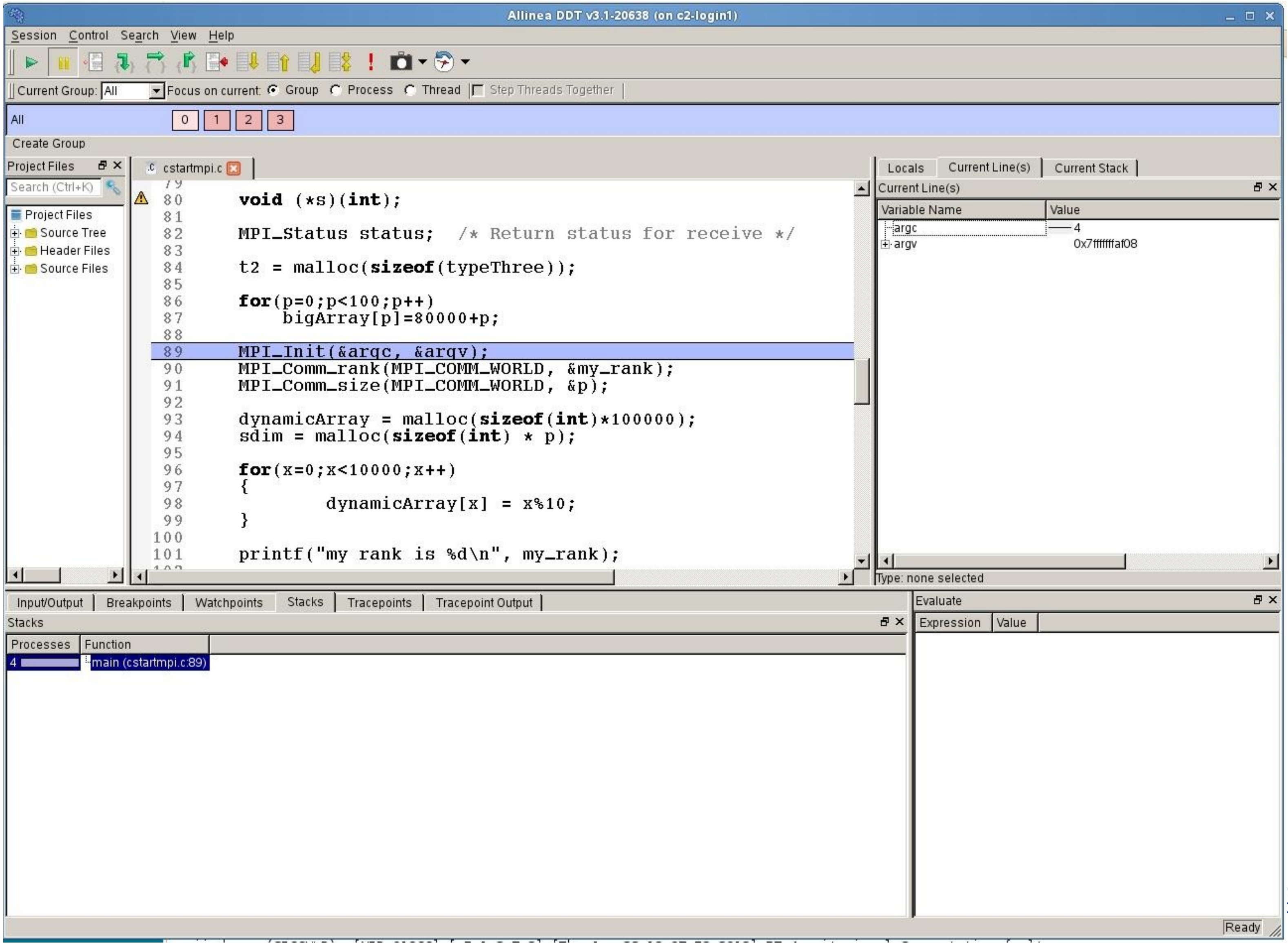
☐ CUDA

☐ **Memory Debugging**

Environment Variables: DMALLOC_OPTIONS debug=0x008

Plugins: none

Source Window



Current Group: All
Focus on current: ● Group ○ Process ○ Thread
□ Step Threads Together

```
        void (*s)(int);

        MPI_Status status;  /* Return status for receive */

        t2 = malloc(sizeof(typeThree));

        for(p=0;p<100;p++)
            bigArray[p]=80000+p;

        MPI_Init(&argc, &argv);
        MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
        MPI_Comm_size(MPI_COMM_WORLD, &p);

        dynamicArray = malloc(sizeof(int)*100000);
        sdim = malloc(sizeof(int) * p);

        for(x=0;x<10000;x++)
        {
                dynamicArray[x] = x%10;
        }

        printf("my rank is %d\n", my_rank);
```

Current Line(s)

Variable Name	Value
argc	4
argv	0x7fffffffaf08

Type: none selected

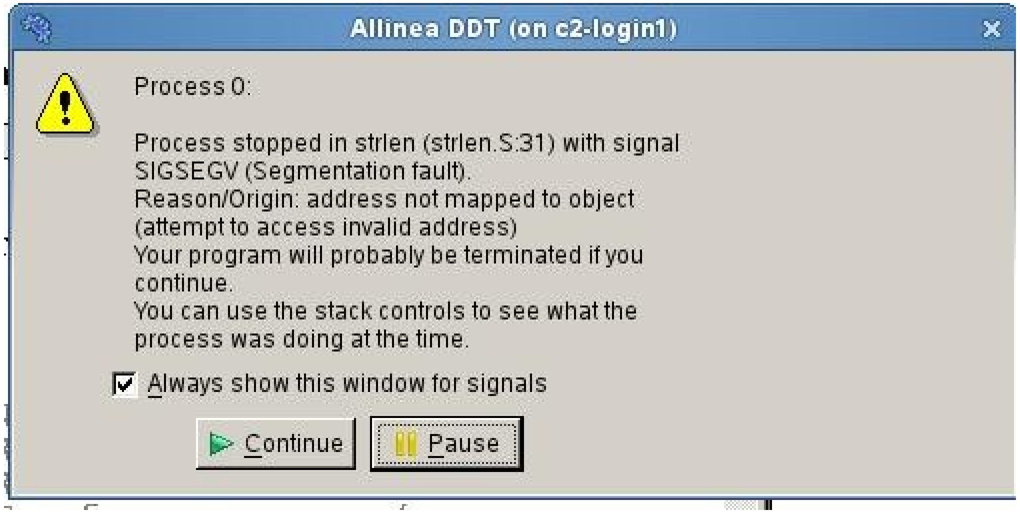
Stacks

Processes	Function
4	main (cstartmpi.c:89)

Ready

SIGSEGV error message

There is an error



Process 0:

Process stopped in strlen (strlen.S:31) with signal SIGSEGV (Segmentation fault).
Reason/Origin: address not mapped to object (attempt to access invalid address)
Your program will probably be terminated if you continue.
You can use the stack controls to see what the process was doing at the time.

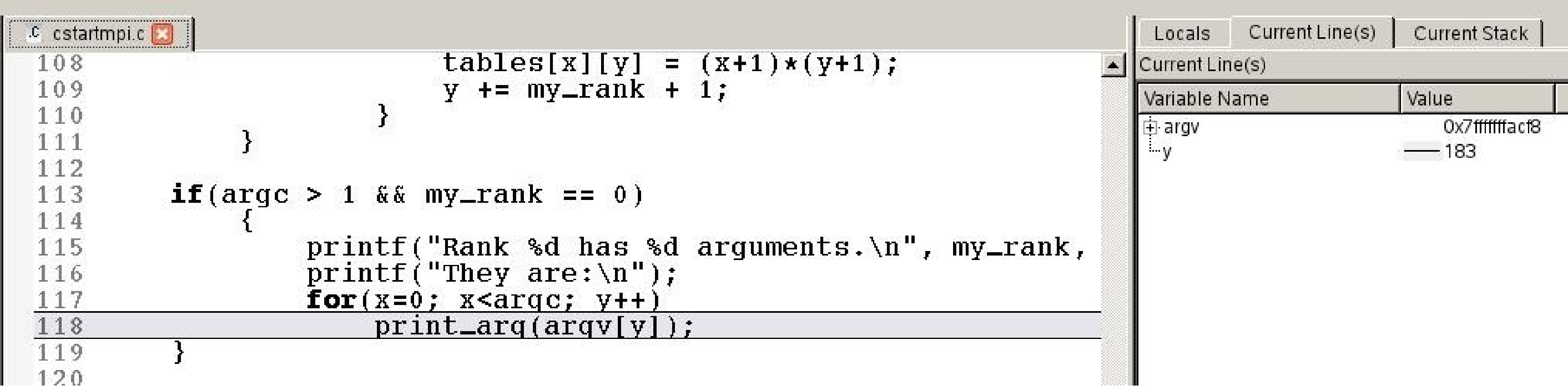
☑ Always show this window for signals

Press Pause

Look at the “Current Stack” window

Current Line Tab

Click on the "Current Line(s)" tab. What is the value of y?



```
                    tables[x][y] = (x+1)*(y+1);
                    y += my_rank + 1;
                }
        }

    if(argc > 1 && my_rank == 0)
        {
            printf("Rank %d has %d arguments.\n", my_rank,
            printf("They are:\n");
            for(x=0; x<argc; y++)
                print_arg(argv[y]);
    }
```

Current Line(s)

Variable Name	Value
argv	0x7fffffffacf8
y	183

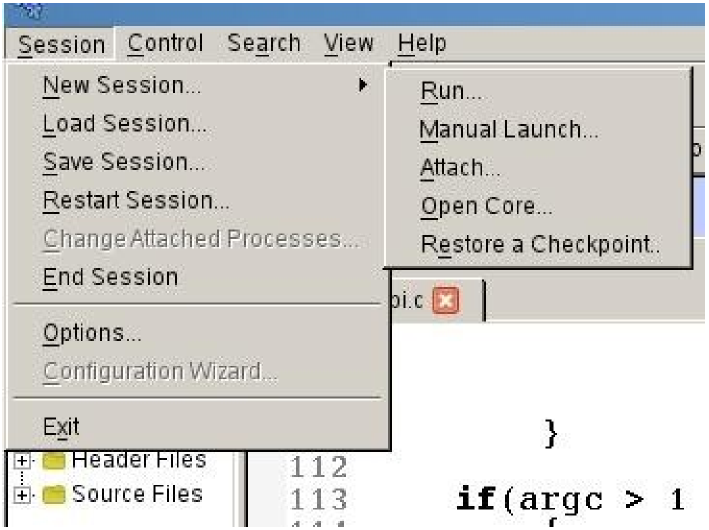
Is the value of y correct?

Is there anything wrong with this loop?

New Session

Correct the loop and run ./run_cstartmpi.csh again to recompile the code.

Session-> New Session-> Run



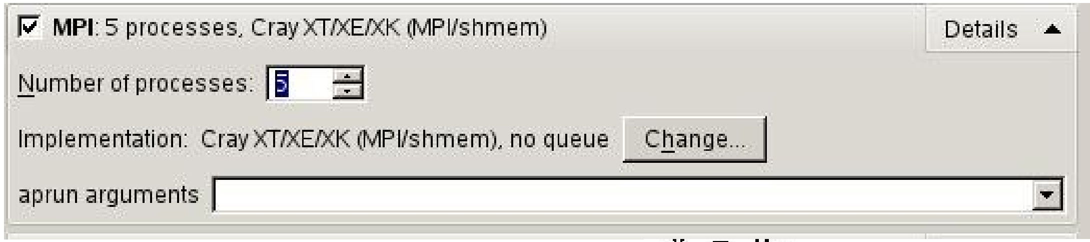
```
112           }
113           if(argc > 1
```

Run the code to completion

Exercise 2

Session-> New Session-> Run

Change the core count to > 4



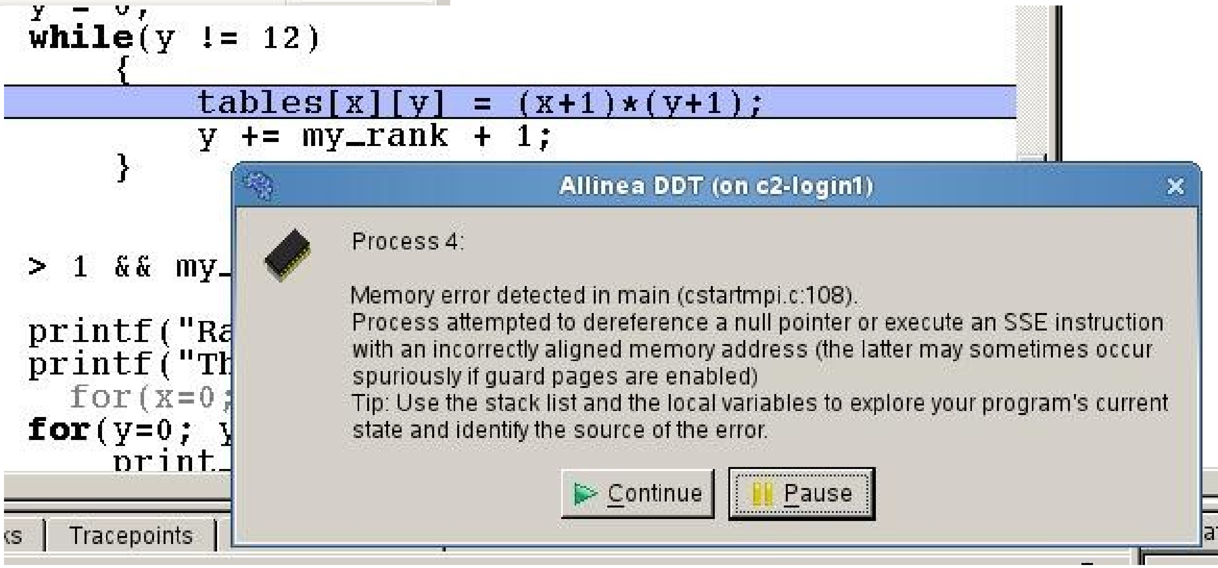
☑ MPI: 5 processes, Cray XT/XE/XK (MPI/shmem)

Number of processes: 5

Implementation: Cray XT/XE/XK (MPI/shmem), no queue

aprun arguments

Find out what is the problem.



```
 y = 0;
while(y != 12)
    {
        tables[x][y] = (x+1)*(y+1);
        y += my_rank + 1;
    }

> 1 && my_

printf("Ra
printf("Th
   for(x=0;
for(y=0; y
    print
```

Allinea DDT (on c2-login1)

Process 4:

Memory error detected in main (cstartmpi.c:108).
Process attempted to dereference a null pointer or execute an SSE instruction with an incorrectly aligned memory address (the latter may sometimes occur spuriously if guard pages are enabled)
Tip: Use the stack list and the local variables to explore your program's current state and identify the source of the error.

trisol exercise

```
cd trisol
./run_trisol.csh
```

Runs program trisol.exe

- Memory error
- Walkthrough of how to turn on memory debugging.

trisol exercise

```
> aprun -n 4 ./trisol.exe
***  Solution correct
|x| / (sqrt(n)*epsilon*(|A|*|x| + |b|) =  3.7352-215
```

```
> ddt -n 4 ./trisol.exe
```

Runs to completion.

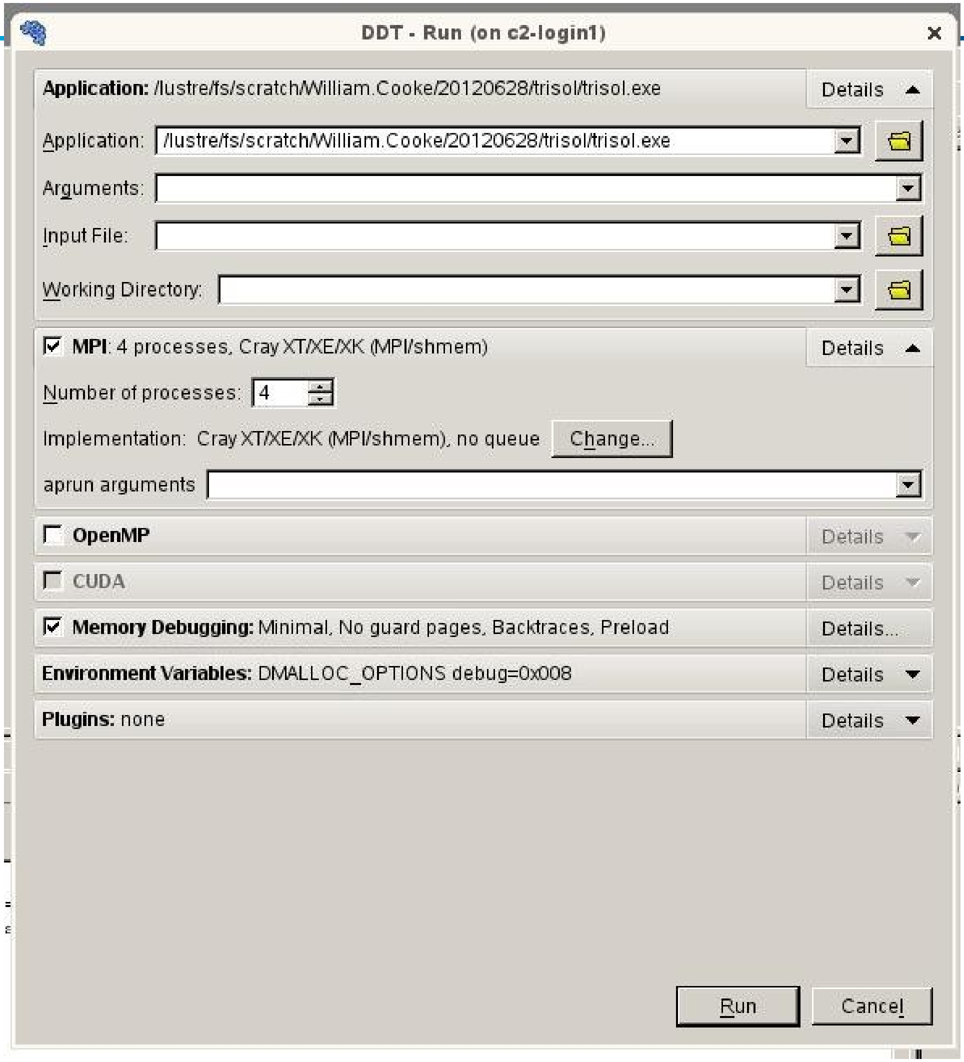
So everything is good?? No. There is a memory error in here.

Memory Debugging

Restart the session.

Check the Memory Debugging box.

Click on Details.



DDT - Run (on c2-login1)

Application: /lustre/fs/scratch/William.Cooke/20120628/trisol/trisol.exe

Application: /lustre/fs/scratch/William.Cooke/20120628/trisol/trisol.exe

Arguments:

Input File:

Working Directory:

☑ **MPI**: 4 processes, Cray XT/XE/XK (MPI/shmem)

Number of processes: 4

Implementation: Cray XT/XE/XK (MPI/shmem), no queue

aprun arguments

☐ **OpenMP**

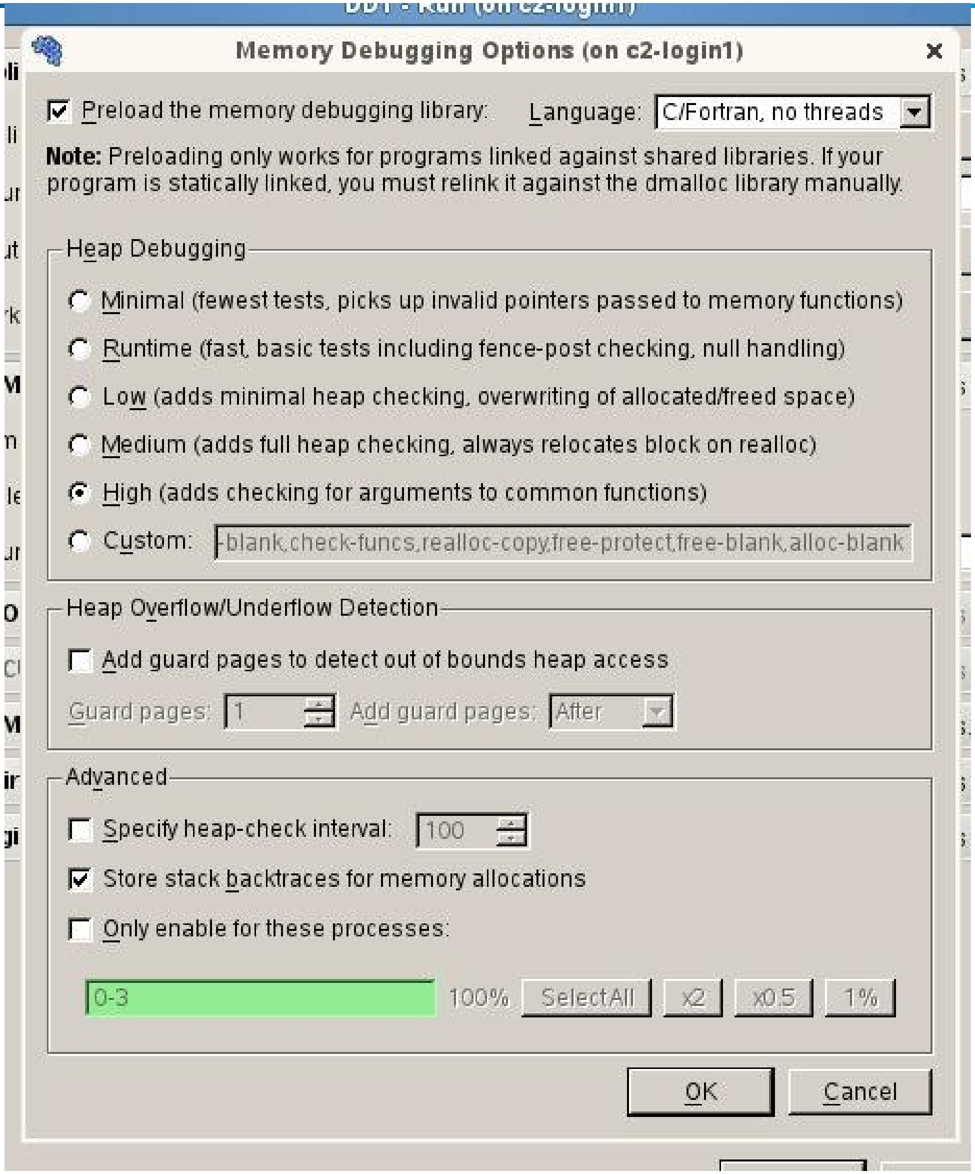
☐ CUDA

☑ **Memory Debugging:** Minimal, No guard pages, Backtraces, Preload

Environment Variables: DMALLOC_OPTIONS debug=0x008

Plugins: none

Memory Debugging Options



Memory Debugging Options (on c2-login1)

☑ Preload the memory debugging library: Language: C/Fortran, no threads

Note: Preloading only works for programs linked against shared libraries. If your program is statically linked, you must relink it against the dmalloc library manually.

Heap Debugging

- ○ Minimal (fewest tests, picks up invalid pointers passed to memory functions)
- ○ Runtime (fast, basic tests including fence-post checking, null handling)
- ○ Low (adds minimal heap checking, overwriting of allocated/freed space)
- ○ Medium (adds full heap checking, always relocates block on realloc)
- ● High (adds checking for arguments to common functions)
- ○ Custom: blank,check-funcs,realloc-copy,free-protect,free-blank,alloc-blank

Heap Overflow/Underflow Detection

□ Add guard pages to detect out of bounds heap access

Guard pages: 1 Add guard pages: After

Advanced

□ Specify heap-check interval: 100

☑ Store stack backtraces for memory allocations

□ Only enable for these processes:

0-3 100%

Check Preload the memory.

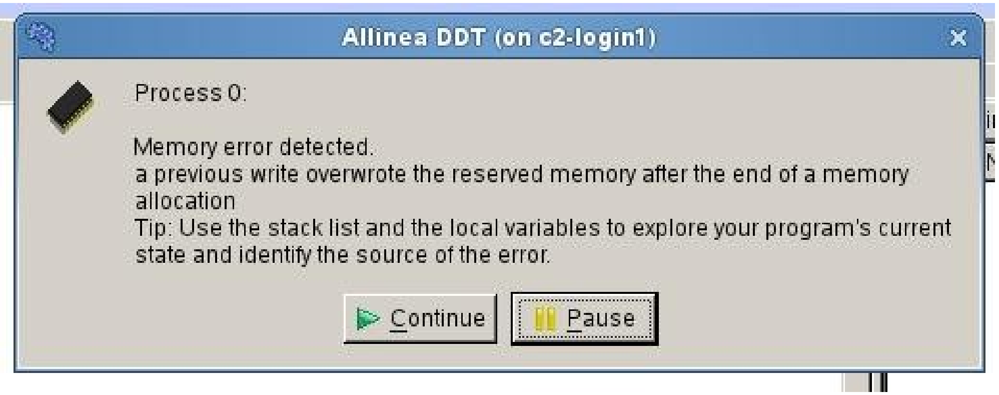
In the Heap debugging options check the High box.

Leave the Heap Overflow unchecked (for now).

Memory Error.

Run the program.

DDT reports an array overflow.



Process 0:

Memory error detected.
a previous write overwrote the reserved memory after the end of a memory allocation
Tip: Use the stack list and the local variables to explore your program's current state and identify the source of the error.

Where is the overflow?

We'll use guard pages to find out.

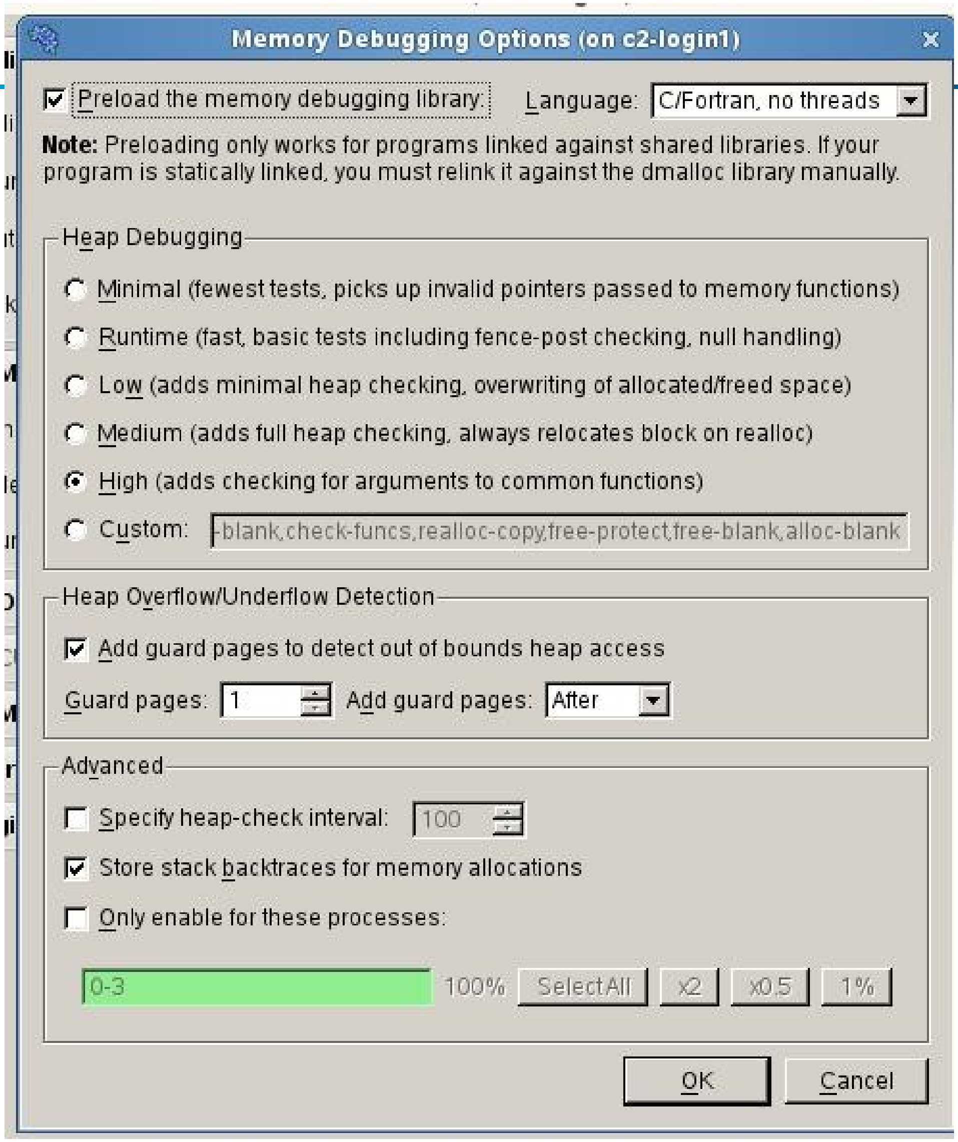
Guard Pages

Restart the session.

Click on Memory Debugging->Details.

Check the "Add guard pages" box under "Heap Overflow/Underflow Detection"

Leave at "1" and "After"



Memory Debugging Options (on c2-login1)

☑ Preload the memory debugging library: Language: C/Fortran, no threads ▾

Note: Preloading only works for programs linked against shared libraries. If your program is statically linked, you must relink it against the dmalloc library manually.

Heap Debugging

- ○ Minimal (fewest tests, picks up invalid pointers passed to memory functions)
- ○ Runtime (fast, basic tests including fence-post checking, null handling)
- ○ Low (adds minimal heap checking, overwriting of allocated/freed space)
- ○ Medium (adds full heap checking, always relocates block on realloc)
- ● High (adds checking for arguments to common functions)
- ○ Custom: -blank,check-funcs,realloc-copy,free-protect,free-blank,alloc-blank

Heap Overflow/Underflow Detection

☑ Add guard pages to detect out of bounds heap access

Guard pages: 1   Add guard pages: After ▾

Advanced

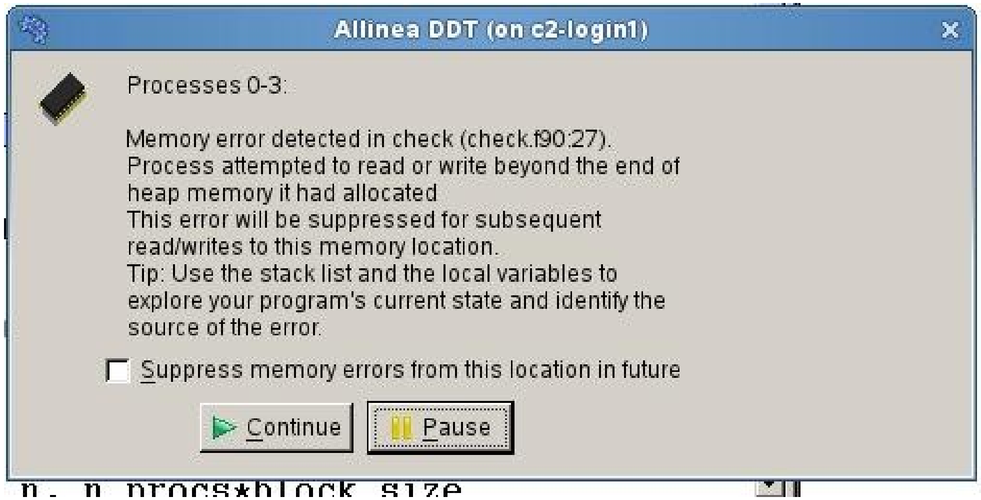
- ☐ Specify heap-check interval: 100
- ☑ Store stack backtraces for memory allocations
- ☐ Only enable for these processes:

0-3   100%

Memory Error.

Run the program again.

DDT reports a memory error in check.f90



Processes 0-3:

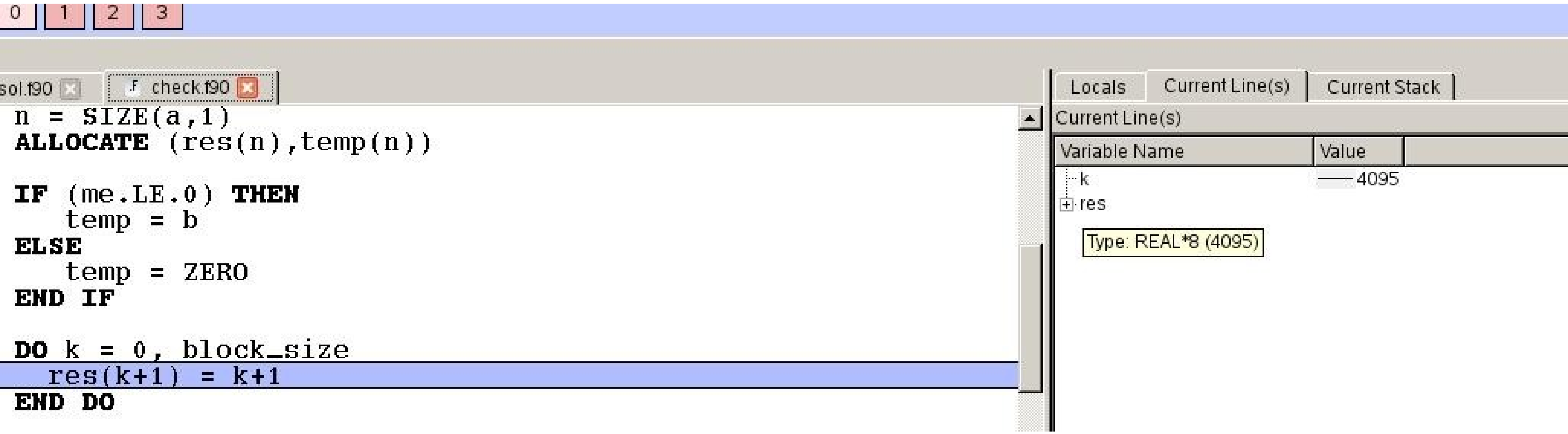
Memory error detected in check (check.f90:27).
Process attempted to read or write beyond the end of heap memory it had allocated
This error will be suppressed for subsequent read/writes to this memory location.
Tip: Use the stack list and the local variables to explore your program's current state and identify the source of the error.

□ Suppress memory errors from this location in future

Click Pause

Look at the size of res in the "Current Lines" tab.

Size of res.



```
n = SIZE(a,1)
ALLOCATE (res(n),temp(n))

IF (me.LE.0) THEN
    temp = b
ELSE
    temp = ZERO
END IF

DO k = 0, block_size
  res(k+1) = k+1
END DO
```

Variable Name	Value
k	4095
res	

Type: REAL*8 (4095)

res is size 4095

k = 4095 so we are trying to write beyond the end of res (res(k+1))

Exercise 3.

Fix the code by inserting

```
DO k = 1, block_size
  res(k) = k
END DO
```

And run ./run_trisol.csh to recompile

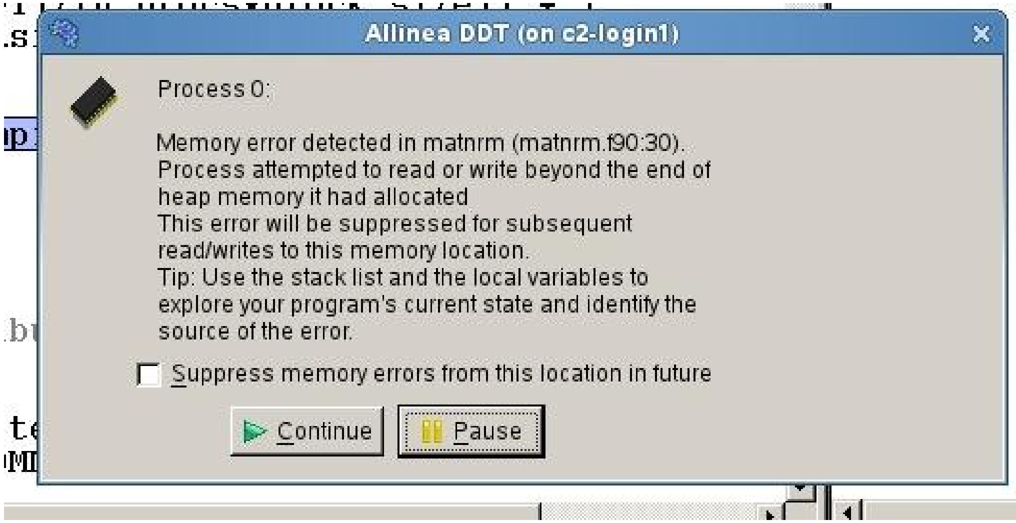
Then

```
ddt -n 4 ./trisol.exe
```

Exercise 3 continued.

You should get a Memory Error in matnrm.

Pause the job and examine values of i, k, j1, j2 on different processors.



Process 0:

Memory error detected in matnrm (matnrm.f90:30).
Process attempted to read or write beyond the end of heap memory it had allocated
This error will be suppressed for subsequent read/writes to this memory location.
Tip: Use the stack list and the local variables to explore your program's current state and identify the source of the error.

□ Suppress memory errors from this location in future

You'll need to pause all cores.

See if you can figure out what is wrong.

Exercise 3 Hints

What is the block_size on each core?

Does it seem correct?

Where is it defined?

How do you define an array in Fortran?

Questions?